GameDevelopers® Conference 07

Game Developers Conference 2007

AMD Smarter Choice

Massive Multi-Core Processors & Gaming

Justin Boggs
Sr. Developer Relations Engineer
justin.boggs@amd.com

Things are about to get real interesting for PC Gaming!

- A Closer Look at Parallelism
- The Berkeley View
- AMD Multi-Core & Optimization
- Optimizing for SSE128
- Software Tools
- Call to Action
- Q&A

AMD
Smarter Choice

A Closer Look at Parallelism

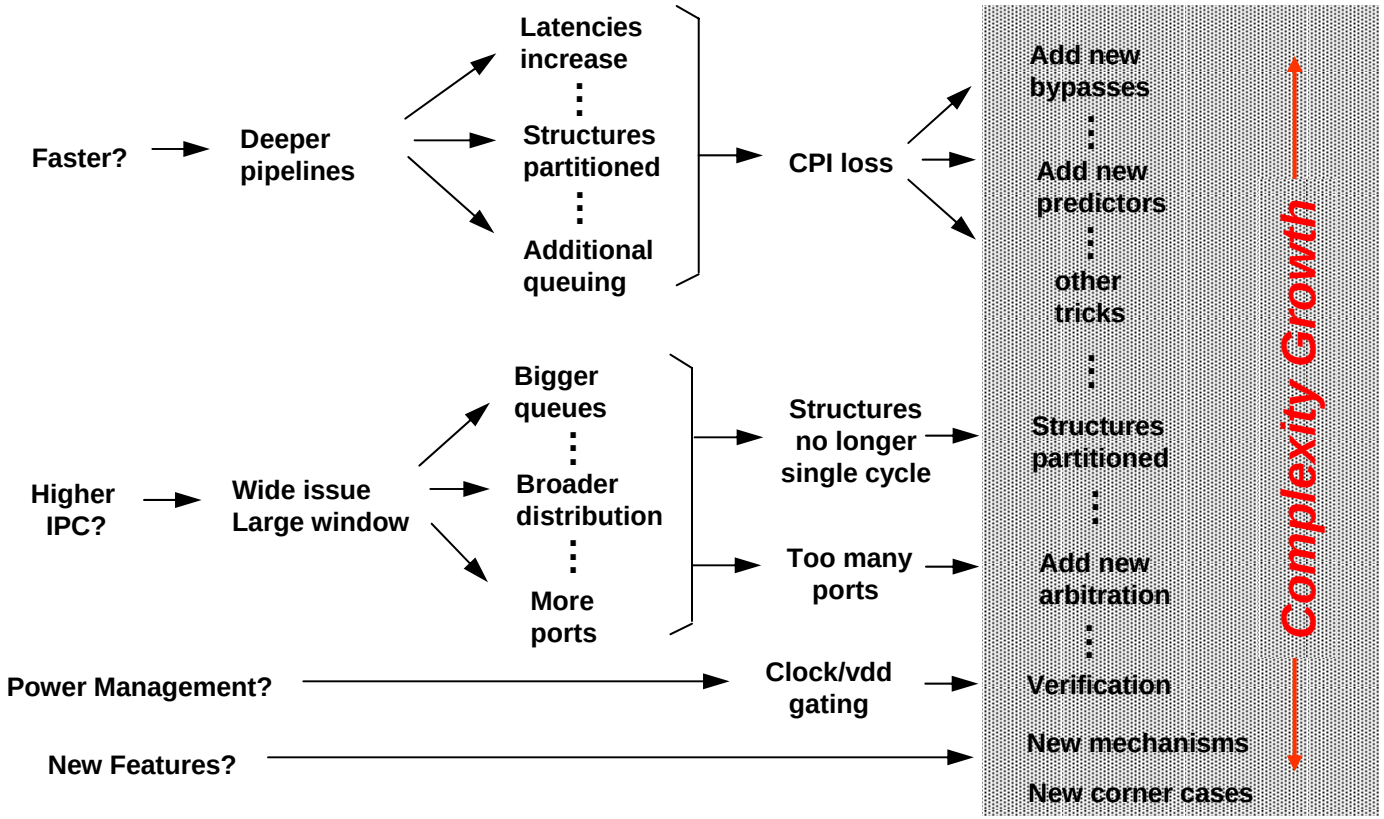
A Closer Look at Parallelism

AMD Smarter Choice

Instruction-level Parallelism (ILP)

Executing multiple instructions from same program at the same time

Superscalar hardware picks up most available ILP (*complexity effective*)



Superposition … *complexity tends to compound!*

A Closer Look at Parallelism

AMD Smarter Choice

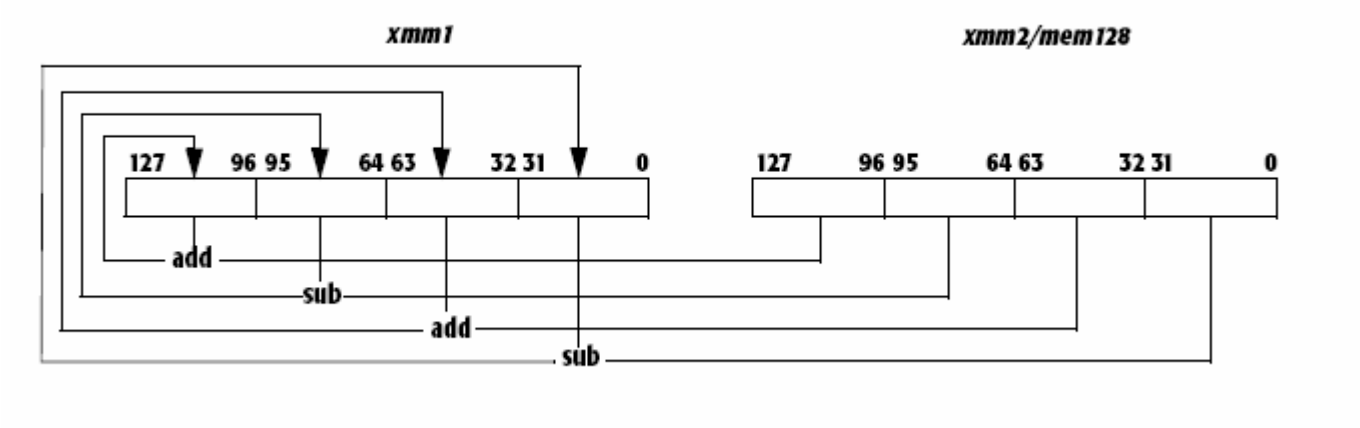
Instruction-level Parallelism (ILP)

Executing multiple instructions from same program at the same time
Superscalar hardware picks up most available ILP (*complexity effective*)

Data-level Parallelism (DLP)

Executing same instruction on multiple pieces of data at the same time
Vector-style processing -- SSE hardware operates in this manner

ADDSUBPS **Add and Subtract Packed Single-Precision**



A Closer Look at Parallelism

AMD Smarter Choice

Instruction-level Parallelism (ILP)

Executing multiple instructions from same program at the same time
Superscalar hardware picks up most available ILP (*complexity effective*)

Data-level Parallelism (DLP)

Executing same instruction on multiple pieces of data at the same time
Vector-style processing -- SSE hardware operates in this manner

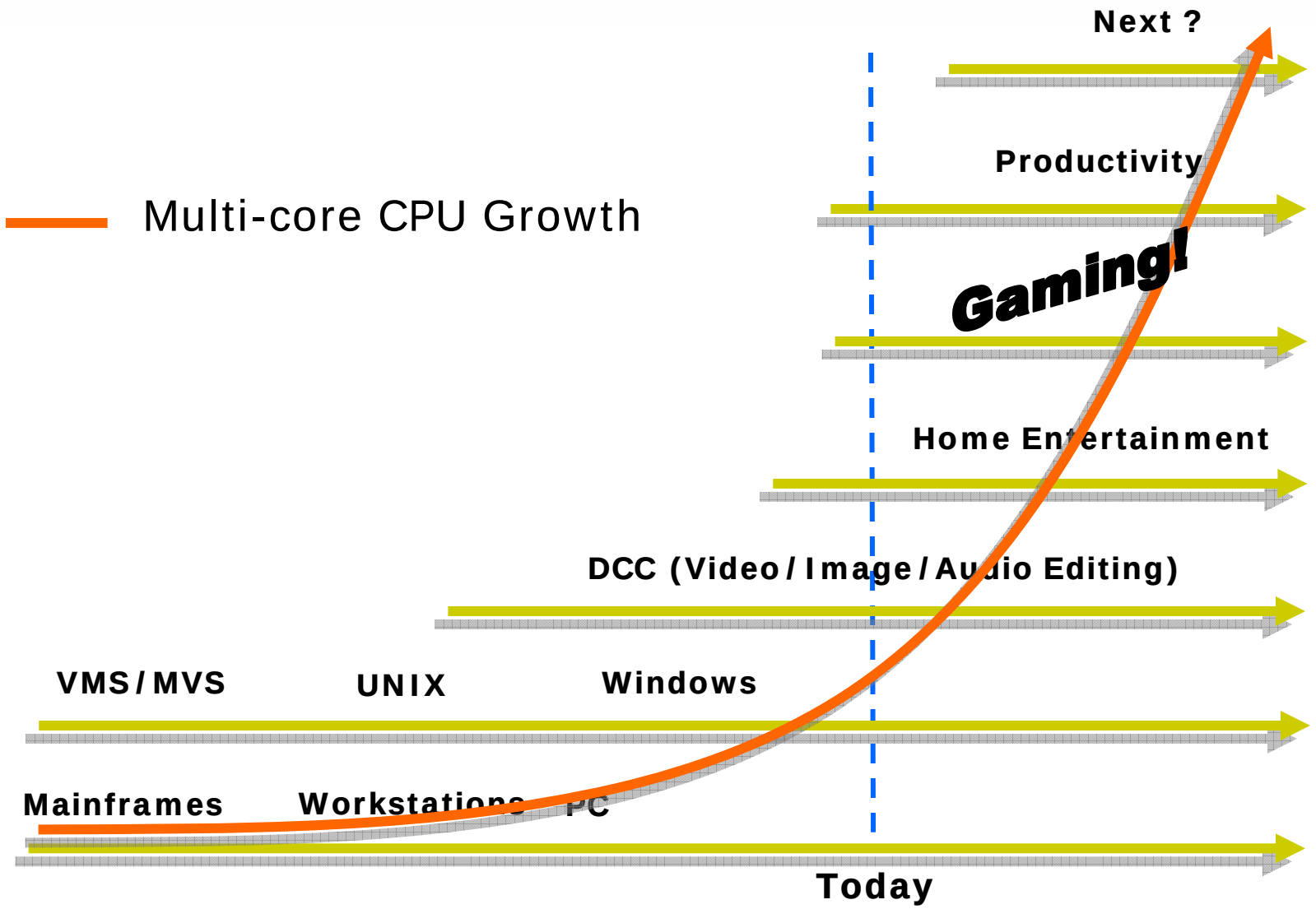
Thread-level Parallelism (TLP) – *several types:*

1. Concurrent Applications
   Multiple programs running at the same time
   Multiple OS's on virtualized hardware image
   Collection of services integrated into a single "application"

2. Internet Transactional
   Multiple computers running the same application

3. Parallel Applications - *The Holy Grail of Computer Science*
   Single application partitioned to run as multiple threads

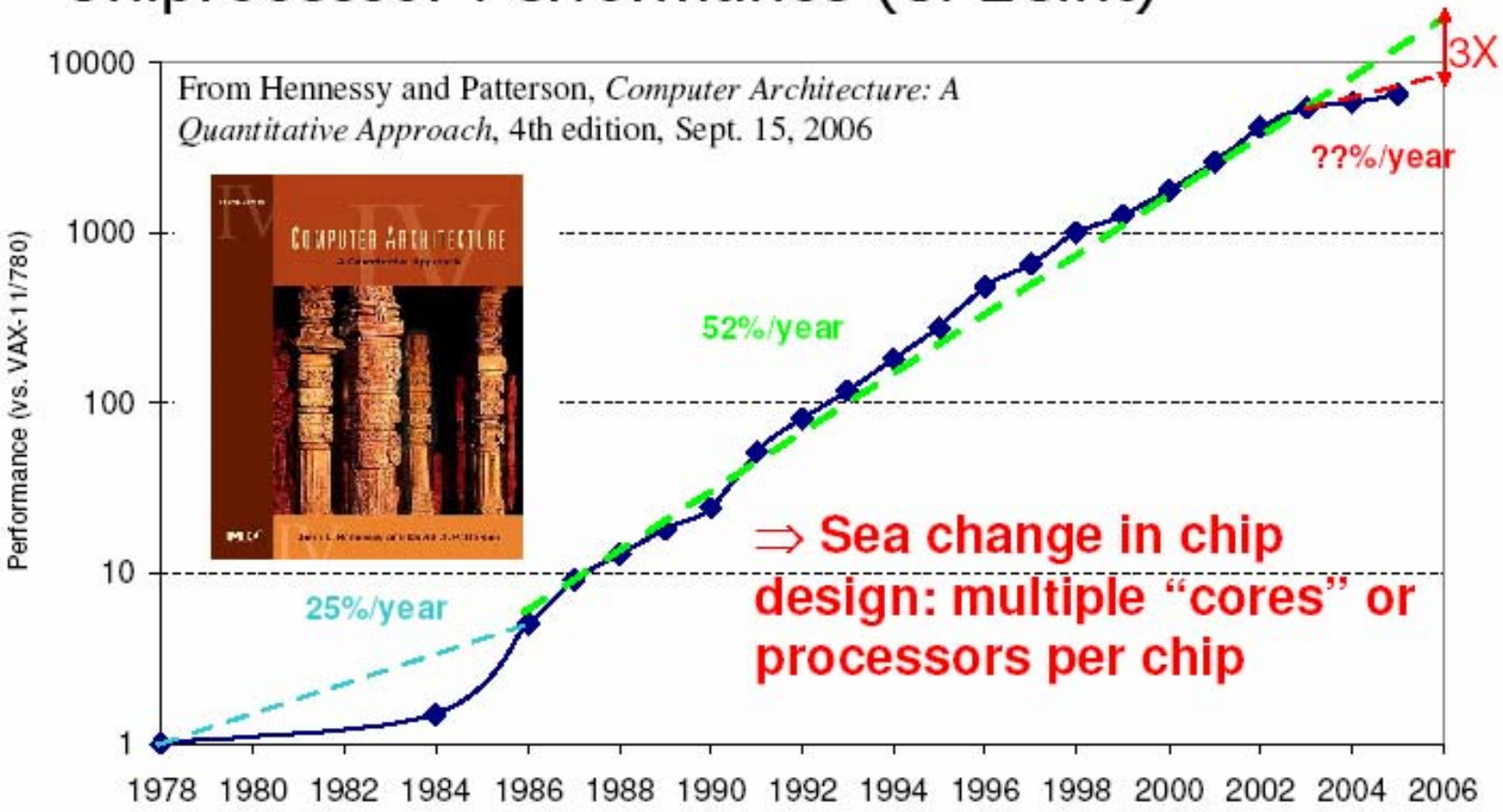
Multi-threaded Workloads



AMD
Smarter Choice

The Berkeley View
(courtesy of Dave Patterson)

Uniprocessor Performance (SPECint)



- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

The Landscape of Parallel Computing Research: A View from Berkeley

AMD Smarter Choice

Old Conventional Wisdom	New Conventional Wisdom
Increasing clock frequency is primary method of performance improvement	Processors Parallelism is primary method of performance improvement
Don't bother parallelizing app, just wait and run on much faster sequential computer	No one is building 1 processor per chip End of La-Z-Boy Programming Era
Less than linear scaling for a multiprocessor is failure	Given the switch to parallel hardware, even sub-linear speedups are beneficial

Published in January 2007 by Krste Asanovic, David A. Patterson, Kurt Keutzer, et al. See http://view.eecs.berkeley.edu

AMD
Smarter Choice

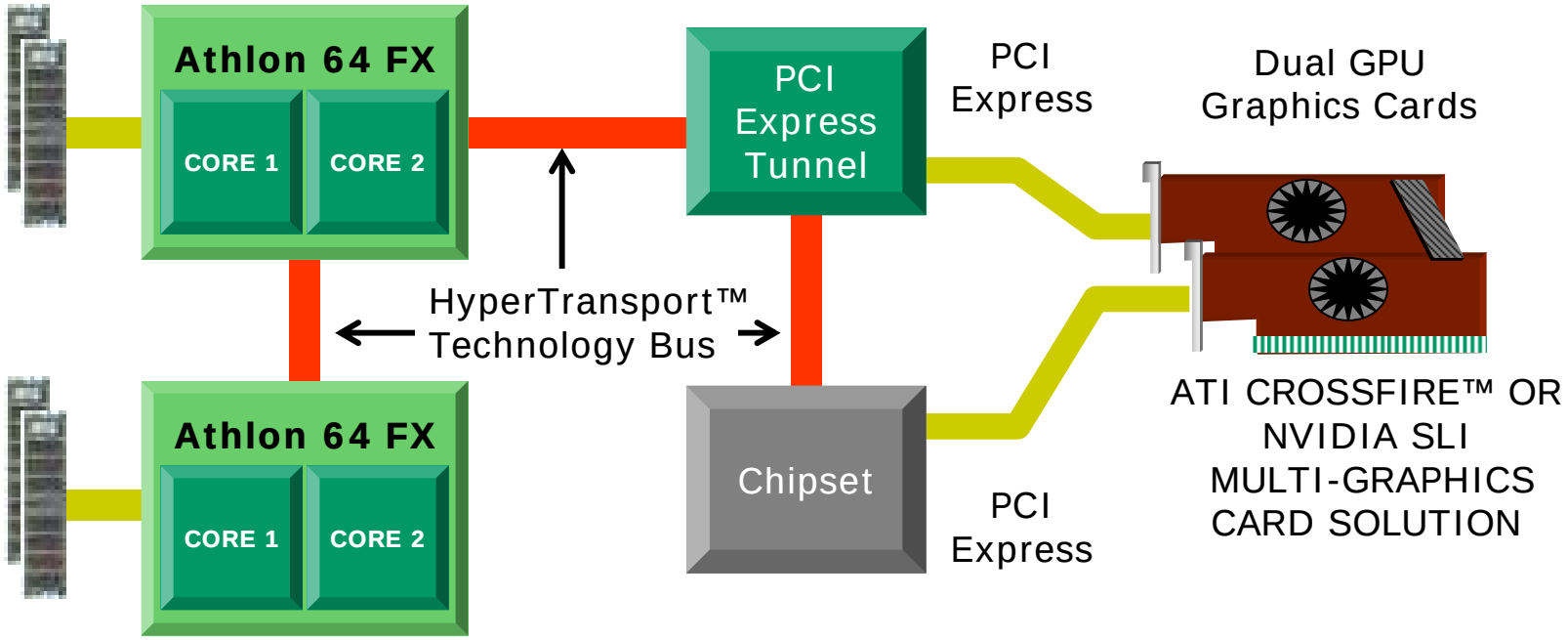
AMD Multi-Core & Optimization

The Quad FX Platform

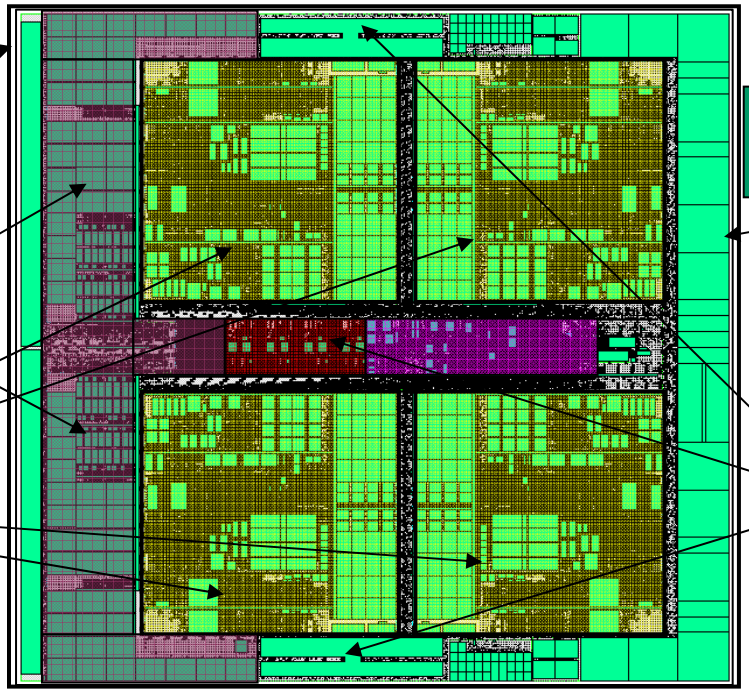
AMD Smarter Choice

The Development Platform for Game Developers

Upgradeable to an 8-core System in 2H'07!



AMD's Quad-Core "Barcelona"



True quad-core die

Expandable shared L3 cache

IPC enhanced CPU cores

Optimum for 65nm SOI and beyond

Enhanced Direct Connect Architecture and Northbridge

- **32-byte instruction fetch**
- **Enhanced branch prediction**
- **Out-of-order load execution**
- **128-bit SSE processing (single-cycle throughput)**
- **Up to 4 double-precision ops per cycle (2 add, 2 mul)**
- **Up to 8 single-precision ops per cycle (4 add, 4 mul)**
- **Dual 128-bit loads per cycle**
- **Bit Manipulation extensions (LZCNT/POPCNT)**
- **SSE extensions (EXTRQ/INSERTQ, MOVNTSD/MOVNTSS)**

- **HyperTransport 3.0 links**
- **Enhanced crossbar design**
- **Enhanced AMD PowerNow!™ Technology**
- **DDR2 support**

Optimizing for cache

AMD Smarter Choice

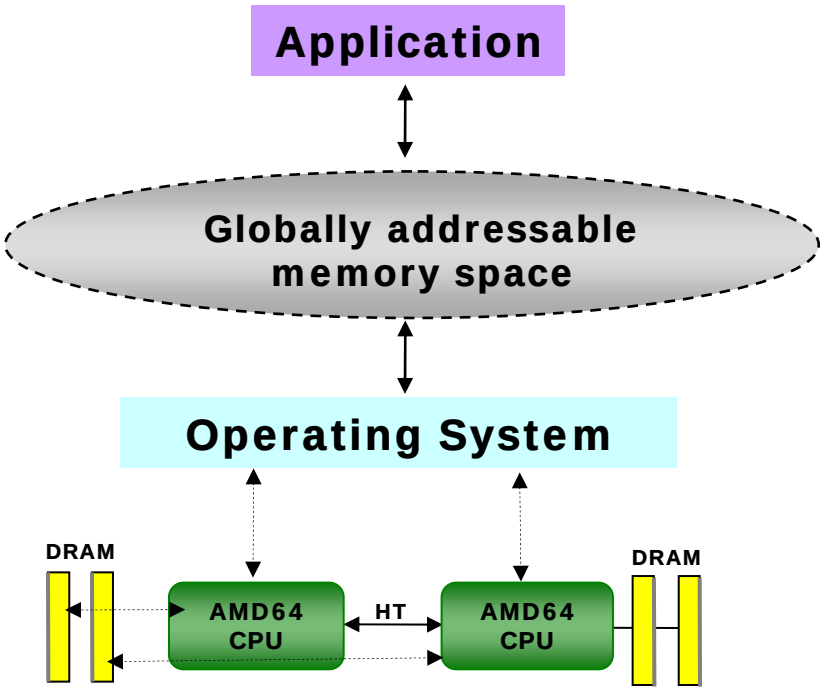
- Efficiently using the cache is important for performance
  - Each core has its own 512k L2 cache
  - L2 cache is “unified”, it stores both instruction and data
- Compiler code generation option
  - Evaluate “minimize code size” instead of “maximize speed”
- Data structures
  - Be careful to avoid wasteful padding in structs (check “sizeof”)
  - Sometimes a 2-byte offset value or array index can replace a 4/8-byte pointer
- Algorithms
  - Process data in blocks of several K-bytes, not megabytes
    Read them using non-temporal software prefetch when appropriate
  - Avoid writing large data sets to memory, then reloading them!
    Keep operating within cache whenever possible
  - Write final data blocks out to memory using Streaming Store
    Avoid disturbing data and code that are in the L2 cache
- Analysis
  - Use AMD CodeAnalyst to measure various cache miss events
  - Easily locate performance trouble spots in your code

Multi-Threading Optimization

AMD Smarter Choice

- Dual-core or greater will be everywhere
  - Utilize those cores!
- Functional threading will run out of steam…
  - Solution: Data Parallel Threads
  - Use OpenMP, thread pools, or your own threads using OS API
- Keep locks to a minimum
  - Take locks as late as possible, release as early as possible
- Beware of false sharing
  - Keep variables updated by different threads on separate 64B cache lines

Non-Uniform Memory Access (NUMA)



Memory initialized into globally addressable memory space with processors maintaining cache coherency across this space

OS assigns threads from same process to the same NUMA node

Processor has local memory and uses HyperTransport™ technology for high-speed access to non-local memory

Multi-Threading Optimization

AMD Smarter Choice

- NUMA awareness
  - Two socket AMD platforms are NUMA machines (ex: Quad FX)
    - *Each socket has a memory controller with local memory*
  - Use OS APIs to determine hardware topology
    - *On XP/Windows Server: `GetNumaNodeProcessorMask()`*
    - *On Vista and "Longhorn": `GetLogicalProcessorInformation()`*
  - Try to keep memory accesses local if multiple threads read/write the same data
    - *Reduces traffic over HyperTransport link*
  - Windows Vista™ and Windows® XP have slightly different memory allocator behaviors
    - *XP allocates memory on the node where the thread is running when memory is first touched (could be remote)*
    - *Vista will allocate memory on a "preferred" node, not on the node that first touches the memory (tries to keep memory local)*
    - *Vista's thread scheduler is also NUMA aware*
    - *See "NUMA optimization in Windows Applications" at http://developer.amd.com/articlex.jsp?id=106 for a code example*

The Cache and Multi-Threading

AMD Smarter Choice

- The shared L3 cache can help threads share data
  - If you are using a producer/consumer thread model…
    - *Keep producer and consumer threads on the same chip*
    - *Set affinity to the same socket (currently = same NUMA node)*
  - Unlike quad-core processors from other vendors, AMD's cache is shared across all cores
- Be careful to avoid "thrashing" the L2 cache
  - Don't have multiple threads modifying the same variables
    - *And don't have one thread modifying it, while others read it*
    - *You can't avoid this if you're using semaphores to sync threads*
  - Don't split a cache line between two threads
    - *Allocate thread-specific memory with 64-byte granularity*
  - CodeAnalyst can easily find these type of trouble spots
    - *Excessive cache miss events*
- Shared cache considerations
  - Heavy cache usage by a thread on one core could negatively impact other threads

Detecting the # of Cores Can Be Tricky

AMD Smarter Choice

- Anticipate the adoption of virtual environments
  - Physical computer may be partitioned: you don't get the whole thing
  - Trust the OS to tell you about available resources

    Really try to avoid using low-level instructions on the hardware

- For a recent OS (Vista and later) use the new function **`GetLogicalProcessorInformation()`**
  - Gives you more complete info about NUMA, cache, etc.
  - Lets you distinguish true cores from SMT threads (pseudo-core)

- For Windows XP / 2000 **`GetSystemInfo()`** tells you how many processors you have

Multi-Threading Resources

- AMD Developer Central – Multi-Core Zone
  http://developer.amd.com/resourcecenter_mc.jsp
- "Designing & Optimizing for N Cores" presentation
  http://developer.amd.com/assets/AMD_Designing_for_N_cores.pdf
- MSDN is a great technical resource
  http://msdn.microsoft.com/
  http://msdn2.microsoft.com/en-us/library/172d2hhw(VS.80).aspx

AMD
Smarter Choice

Optimizing for SSE128

SSE128 improves performance

- Doubles VECTOR SSE performance vs. previous CPUs
- Example : ADDPD (Add Packed Double)
  - AMD eighth generation processor e.g. current AMD Athlon™ 64
    - *Decodes into two ADD micro-ops*
    - *Each micro-op takes a pass through the 64-bit adder*
  - "Barcelona" w/ SSE128
    - *Decodes into a single ADD micro-op*
    - *Takes one pass through the 128-bit adder*
- Example performance gains
  - Matrix multiply performance improves by ~85%
  - Other math intensive apps show 10%-50% gains
- **Vectorized SSE is very fast with SSE128**
  - Much faster than x87 & scalar SSE
  - Double-precision is 2x faster than x87
  - Single-precision is 4x faster than x87

Comprehensive Upgrades for SSE128

Current Generation versus Next Generation

AMD Smarter Choice

Parameter	Current Processor	"Barcelona"
SSE Exec Width	64	128 + SSE MOVs
Instruction Fetch Bandwidth	16 bytes/cycle	32 bytes/cycle + Unaligned Ld-Ops
Data Cache Bandwidth	2 x 64bit loads/cycle	2 x 128bit loads/cycle
L2/NB Bandwidth	64 bits/cycle	128 bits/cycle
FP Scheduler Depth	36 Dedicated x 64-bit ops	36 Dedicated x 128-bit ops

- Can perform SSE MOVs in the FP "store" pipe
  - Execute two generic SSE ops + SSE MOV each cycle (+ two 128-bit SSE loads)
- SSE Unaligned Load-Execute mode
  - Remove alignment requirements for SSE load-op instructions
  - Eliminate awkward pairs of separate load and compute instructions
  - *To improve instruction packing and decoding efficiency*

SSE128 Caveats

AMD Smarter Choice

- Only *vectorized* SSE is accelerated
  - No improvement for scalar code
- SSE128 can't separately address 64-bit halves of XMM reg
  - MOVLPD/MOVHPD internally need to merge before writing 128-bit register
  - Merge creates an extra dependency on output register:

    Example: MOVLPD XMM0, […]

    SSE128 part will read XMM0 ; today's parts just overwrite XMM0L
  - Impacts these instructions:

    movhpd, movhps

    movlpd, movlps

    movsd *(reg version only)*

    cvtpi2ps, cvtsi2sd, cvtss2sd

    sqrtsd

    movhlps, movlhps
- SSE128 adds an additional register read pipe stage
  - Only impacts floating point pipeline
  - Adds a cycle to FP load latency

SSE128 Loads and Stores

AMD Smarter Choice

- Data cache bandwidth
  - Two ports (banked)
  - Each port supports either a 128-bit load or a 64-bit store
    Better than AMD eighth generation (current) processor, where each port supports either a 64-bit load or a 64-bit store
  - **Software: You may use SSE for copy loops to take advantage of increased data cache load bandwidth**
- Loads are decoded as single 128-bit micro-op
  - Includes MOVUPD/MOVUPS/MOVDQU (misaligned loads)
    No penalty for these if aligned
- Stores are decoded into two 64-bit micro-ops
  - But data delivered from floating point to DC in a single 128-bit chunk
- 128-bit Store-To-Load Forwarding
  - Can forward from 128-bit stores to 128-bit loads
  - Not from MOVHPD/MOVLPD stores to 128-bit load

Optimizing your own code for SSE128

AMD Smarter Choice

- Vectorize your own numeric code for best performance
  - Up to 4x performance gain vs. scalar code
- "Blended" optimizations with no performance impact on FP64
  - Be aware of input dependency on MOVLPD/MOVHPD and MOVSD

    Avoid choosing a recently-written register:

    ```
    MULPD   XMM7, ,,,
    MOVLPD  XMM7, [ ]   ;; in SSE128, has dependency on MULPD.
    ```

    Or perhaps replace MOVLPD-mem with MOVSD-mem
  - Schedule MOVLPD/MOVHPD pairs as taking 2 extra cycles
- For best performance on SSE128:
  - Replace MOVLPD / MOVHPD pairs with MOVUPD or MOVDDUP
  - Replace MOVLPD-mem with MOVSD-mem (upper 64 bits are zeroed)
  - Replace MOVSD-reg with MOVAPD
  - Take advantage of misaligned load-op mode (special code path required)
- Use SHUFPS/SHUFPD instead of unpcklps/hps/lpd/hpd

Optimized libraries for SSE128

AMD Smarter Choice

- Use vectorized SSE libraries!
  - Why reinvent the wheel?
- Use AMD Performance Library, APL
  - Optimized SSE code, vectorized and multi-threaded
  - Functions include array math and image processing
  - http://developer.amd.com/apl.jsp
- Other vector optimized libraries
  - **RAD Game Tools**: Bink, Miles, Granny, and Pixo use SSE extensively
  - **Havok** : SSE vectorized version is ~40% faster vs. the non-SIMD version
  - And many more!

AMD
Smarter Choice

Software Tools

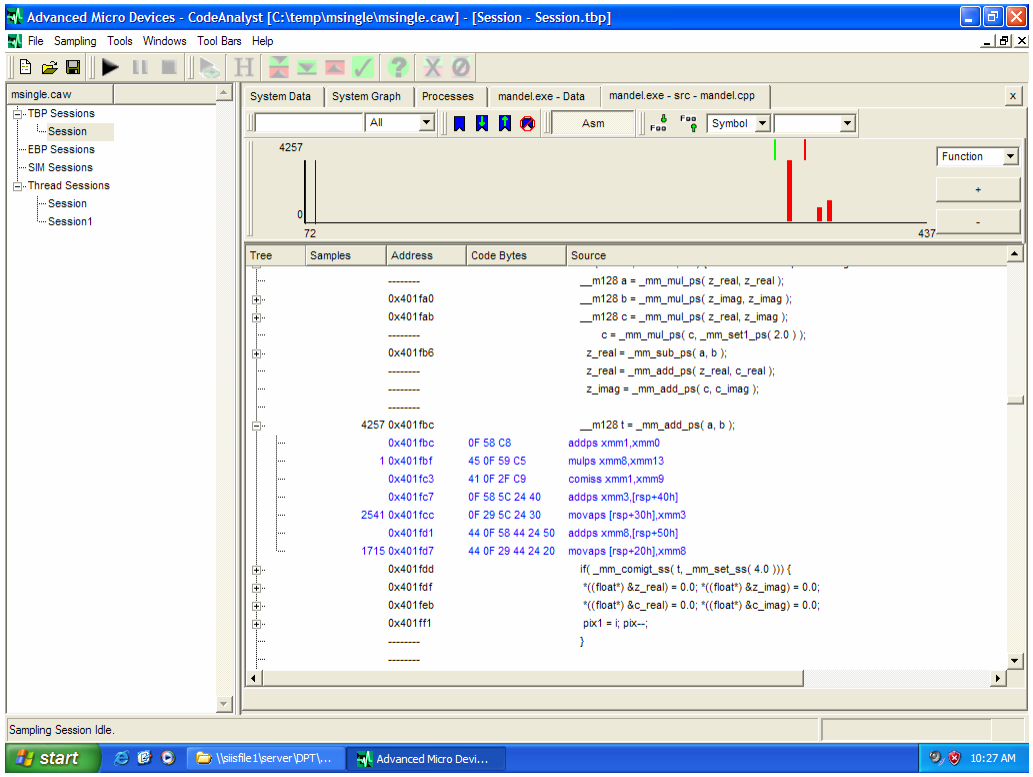
AMD CodeAnalyst™ profiler

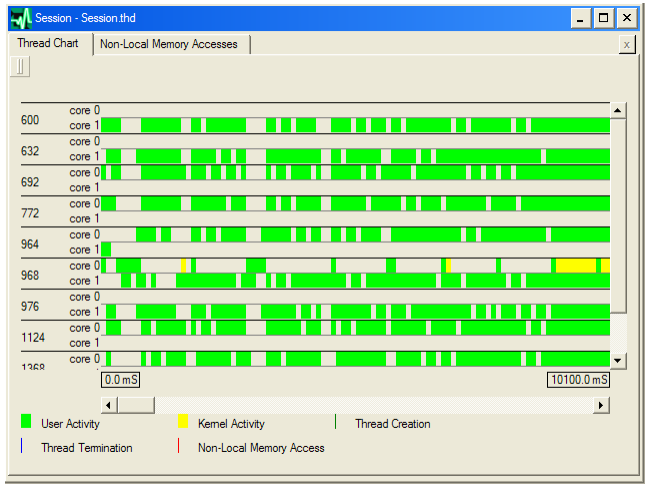
AMD
Smarter Choice

- Profile your code to find hot spots
  - Timer-based sampling (on Intel® CPUs too)
  - Event-based sampling
  - Thread profiler
- Very easy to get started with CodeAnalyst
  - Learn a lot about your application quickly
- Also see system code, drivers, other applications
- 32-bit and 64-bit, Windows® and Linux®
- Download it for free from http://developer.amd.com

AMD CodeAnalyst™ Profiler

- Use AMD CodeAnalyst, profile your 32 and 64-bit code!
  - Timer-based & event-based profiler
  - Integrates with the VS2005 IDE





- *Thread* Profiler shows a thread execution timeline

APL Overview

The AMD Performance Library (APL) is a collection of software routines designed to help developers write high performance multimedia software.

- ***Simplifies*** application development, debugging, and optimization
- Provides ***future proof*** interfaces to the processor; as processors evolve, so does APL, but the API stays the same
- Designed to work on all AMD ***compatible*** processors
- Currently focuses on image and signal processing functions
- Give us your ***feedback*** on what functions you want for ***game development***

http://developer.amd.com/apl.jsp

More information at AMD Dev Central

AMD Smarter Choice

Real-world solutions with practical guidance.

- Detailed technical articles
- Documentation, tutorials, and guides
- Case studies

Free tools and resources to get your job done better, faster.

- CodeAnalyst Performance Analyzer
- AMD Core Math Library
- Multi-core Technology Zone

Expertise from the leader in x86 and 64-bit computing.

- Secure access to latest AMD systems
- Sneak peek at emerging technologies
- Inside look at AMD's vision

Join today, it's free!

developer.amd.com

AMD
Smarter Choice

Call to Action

Call to Action

AMD
Smarter Choice

- Quad-core processors & 8-core desktop systems are here in 2007… the time to multi-thread is now
  - Optimize for Quad FX / NUMA too
- Use Data Parallel Threading as your threading model for scalable performance for years to come
- Vectorize your math intensive code by using optimized libraries (ex: APL) or by writing optimized SSE code
- Test, develop, and optimize on AMD… run fast everywhere!

AMD
Smarter Choice

Q&A

References

• “Computer Architecture is Back - The Berkeley View on the Parallel Computing Landscape” presentation by David Patterson, Krste Asanovíc, Kurt Keutzer, and a cast of thousands, U.C. Berkeley, January 2007

Disclaimer & Attribution

DISCLAIMER

The information presented in this document is for information purposes only. AMD has used commercially reasonable efforts to confirm that the procedures and other information set forth in this document are accurate and reliable. Despite such efforts, the information contained herein may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including, but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN OR FOR THE PERFORMANCE OR OPERATION OF ANY PERSON, INCLUDING, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, DAMAGE TO OR DESTRUCTION OF PROPERTY, OR LOSS OF PROGRAMS OR OTHER DATA, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION